MetaRule

Buffer Management

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-29

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 13134 bytes

Attack Category	• Malicious Input • Denial of Service
Vulnerability Category	• Buffer Management • Buffer Overflow • No Null Termination • Privilege escalation problem • Multibyte Character
Software Context	• String Management
Location	
Description	Many functions are susceptible to buffer management and bounds-checking errors. There are many generic types of errors that can apply to usage of a wide variety of functions. These include: * using a function that does not permit one to specify the size of a buffer to prevent overflows * mis-specifying the size of a buffer or the amount of data to be written. Off-by-one errors are common. * failing to plan for correct behavior when input is larger than expected * failing to allow space for a terminating null character * failing to ensure that a terminating null character is present (many standard functions consistently experience this failure) * specifying the size of a buffer or the amount of data to be transferred using incorrect units. This is particularly a problem with multibyte strings. On the Windows platform, these functions tend to include a "W" in the name. See the rule "MULTIBYTE" for more information. * assuming the wrong semantics for a parameter that controls data transfer and prevents buffer overflows. Because various functions use the buffer size, buffer size minus one, the remaining space in the buffer,

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

	etc., it is important to understand the bounding semantics for each function. Note that while some functions such as strcpy() are intrinsically dangerous, even the "safe" functions like strncpy() are still susceptible to subtle errors if bounds checks are not done properly.
APIs	

Function Name	Comments
_mbsnbcat	
_mbsnbcpy	
_mbsncpy	
_tcsncat	
_tcsncpy	
_tcsxfrm	
bcopy	bytes, not strings
CopyMemory	
fgets	
lstrcpynW	
memcpy	bytes, not strings
sprintf	
snprintf	fmt: 2; src: 3 variable; good use of copying, concat,
StrCatBuff	uses BuffSize, not "max chars to append"
StrCatBuffA	uses BuffSize, not "max chars to append"
StrCatBuffW	uses BuffSize, not "max chars to append"
StrCatN	
StrCatNA	
StrCatNW	
strccat	Solaris, substitute escape chars for binary value

strcpy	
strccpy	Solaris, substitute escape chars for binary value
StrCpyN	"StrCpy" routines are from shell, Shlwapi.dll
StrCpyNA	
StrCpyNW	
StrFormatByteSize	
StrFormatByteSize64	
StrFormatByteSize64A	
StrFormatByteSize64W	
StrFormatByteSizeA	
StrFormatByteSizeW	
StrFormatKBSize	
StrFormatKBSizeA	
StrFormatKBSizeW	
StrFromTimeInterval	
StrFromTimeIntervalA	
StrFromTimeIntervalW	
strncat	have to keep track of size as it builds up
StrNCat	
strncpy	make sure null terminated
strxfrm	
vsnprintf	fmt: 2; src: 3 variable;
wcsncat	
wcsxfrm	
wnsprintf	fmt: 2; src: 3 variable;
wnsprintfA	

wnsprintfW	
wvnsprintf	fmt: 2; src: 3 variable;
wvnsprintfA	
wvnsprintfW	

Method of Attack	Bounds checking, null termination and off-by-one errors create opportunities for buffer overflow or denial of service attacks.
Exception Criteria	

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Always	Search for the functions identified in this rule and ensure that correct bounds checking is done. Be aware that other functions may have similar issues. Pay particular attention to whether it is possible to get an off-by-one error and that the strings are properly terminated with NULL when done.	Effective to the degree that consistent care is used.
Always	Identifying or writing safer versions of utility functions that incorporate checks and then using these safer functions consistently improves safety.	Effective to the degree that consistent care is used.
Always	Always use #define or other const for	Effective to the degree that

		declaration of size.	consistent care is used.
	Always	Always use SAME #define or const when checking bound sizes.	Effective to the degree that consistent care is used.
	Always	For strings, use (buffer size) - 1 to ensure space to put terminating \0.	Effective to the degree that consistent care is used.
	Always	Always write a \0 to upper bound of buffer after processing string.	Effective to the degree that consistent care is used.
	Always	Do a bounds check to verify that the buffer you are passing in is as big as you say and that it is big enough to hold the new contents. Verify that the returned buffer is null terminated.	Effective to the degree that consistent care is used.
	Always	Take particular care when working with multibyte strings.	Effective to the degree that consistent care is used.

Signature Details	Any code with the identified functions.

Examples of Incorrect Code	`char str1[10];`
	`char str2[]="abcdefghijklmn";`
	`strcpy(str1,str2);`

Examples of Corrected Code	`/* If truncation is ok, the following works. */`
	const int BUFFER_SIZE = 10;
	char str1[BUFFER_SIZE];
	char str2[]="abcdefghijklmn";
	/* in this case we know str1 isn't null, but in general we should check to confirm that. */
	`/* strncpy() always works, but on systems such as Windows or BSD Unix, there are better choices. */`

	`strncpy(str1,str2, BUFFER_SIZE-1); /* limit number of characters to be copied */ str1[BUFFER_SIZE-1] = '\0'; / * guarantee result will be null terminated */`
Source Reference	• Viega, John & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X
Recommended Resource	
Discriminant Set	**Operating Systems**: • Any • Windows • UNIX **Languages**: • C • C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com